PeerJ Computer Science

Software development: do good manners matter?

Giuseppe Destefanis[1], Marco Ortu[2], Steve Counsell[1], Stephen Swift[1], Michele Marchesi[2] and Roberto Tonelli[2]

[1] Department of Computer Science, Brunel University, London, United Kingdom
[2] Department of Electrical and Electronic Engineering, University of Cagliari, Cagliari, Italy

ABSTRACT

A successful software project is the result of a complex process involving, above all, people. Developers are the key factors for the success of a software development process, not merely as executors of tasks, but as protagonists and core of the whole development process. This paper investigates social aspects among developers working on software projects developed with the support of Agile tools. We studied 22 open-source software projects developed using the Agile board of the JIRA repository. All comments committed by developers involved in the projects were analyzed and we explored whether the politeness of comments affected the number of developers involved and the time required to fix any given issue. Our results showed that the level of politeness in the communication process among developers does have an effect on the time required to fix issues and, in the majority of the analysed projects, it had a positive correlation with attractiveness of the project to both active and potential developers. The more polite developers were, the less time it took to fix an issue.

Subjects Data Mining and Machine Learning, Data Science, Software Engineering

Keywords Social and human aspects, Politeness, Mining software repositories, Issue fixing time, Software development

INTRODUCTION

High-level software development is a complex activity involving a range of people and activities; ignoring human aspects in the software development process or managing them in an inappropriate way can, potentially, have a huge impact on the software production process and team effectiveness. Increasingly, researchers have tried to quantify and measure how social aspects affect software development. Bill Curtis claimed that "the creation of a large software system must be analyzed as a behavioural process" (*Curtis, Krasner & Iscoe, 1988*). Coordinating and structuring a development team is thus a vital activity for software companies and team dynamics have a direct influence on group successfulness. Open-source development usually involves developers that voluntarily participate in a project by contributing with code-development. In many senses, the management of such developers is more complex than the management of a team within a company: developers are not in the same place at the same time and coordination therefore becomes more difficult. Additionally, the absence of face-to-face communication mandates the use of alternative technologies such as mailing lists, electronic boards or issue tracking systems. In this context, being rude or aggressive when writing a comment or replying to a contributor can affect the cohesion of the group, its membership and the successfulness of a project.

Submitted 17 November 2015
Accepted 15 June 2016
Published 18 July 2016

Corresponding author
Giuseppe Destefanis,
giuseppe.destefanis@brunel.ac.uk

Academic editor
Arie van Deursen

Additional Information and Declarations can be found on page 29

DOI **10.7717/peerj-cs.73**

On the other hand, a respectful environment provides an incentive for new contributors to join the project and could significantly extend the lifetime and usefulness of a project to the community.

According to VersionOne (https://www.versionone.com/pdf/2013-state-of-agile-survey.pdf) (*VersionOne*, *2013*): "more people are recognising that agile development is beneficial to business, with an 11% increase over the last two years in the number of people who claim that agile helps organisations complete projects faster". A main priority reported by users was to accelerate time to market, manage changing priorities more easily and better align IT and business objectives. Agile project management tools and Kanban boards experienced the largest growth in popularity of all agile tool categories, with use or planned use increasing by 6%. One of the top five ranked tools was Atlassian JIRA (https://www.atlassian.com/software/jira), with an 87% recommendation rate. Agile boards represent the central aspect of communication in the Agile philosophy. According to *Perry* (*2008*) "the task board is one of the most important radiators used by an agile team to track their progress." The JIRA board is a good solution for bridging the gap between open-source software development and the Agile world. It is the view of many that agile development requires a physical aspect, i.e., developers working together in the same room or building, or at the same desk; the pair programming paradigm, for example, requires at least two people working simultaneously on the same piece of code. By using tools such as the JIRA board (Fig. 1) it is possible to use an agile board for development of a project by developers in different physical places. Working remotely, in different time zones and with different time schedules, with developers from around the world, requires coordination and communication. The JIRA board displays issues from one or more projects, giving the possibility of viewing, managing and reporting on work in progress. It is possible to use a board that someone else has created, or create as many boards as needed.

When a new developer joins a development team, the better the communication process works, the faster the new developer can become productive and the learning curve reduced. The notion of an agile board therefore places emphasis on the know-how and shared-knowledge of a project being easily accessible for the development team throughout the development process. Fast releases, continuous integration and testing activities are directly connected to the knowledge of the system under development. The potential for agile boards to simplify development across geographically disparate areas is in this sense relatively clear. In a similar vein, the social and human aspects of the development process are becoming more and more important. The Google work style has become a model for many software start-ups: a pleasant work environment is important and affects the productivity of employees.

One important contributor to a healthy work environment is that each employee is considerate and polite towards their fellow employees. Collins dictionary (http://www.collinsdictionary.com/dictionary/english/polite) defines politeness as "showing regard for others, in manners, speech, behaviour, etc." We focus on the politeness of the comment-messages written by the developers. The research aims to show how project management tools such as agile boards can directly affect the productivity of a software development team and the health of a software project.

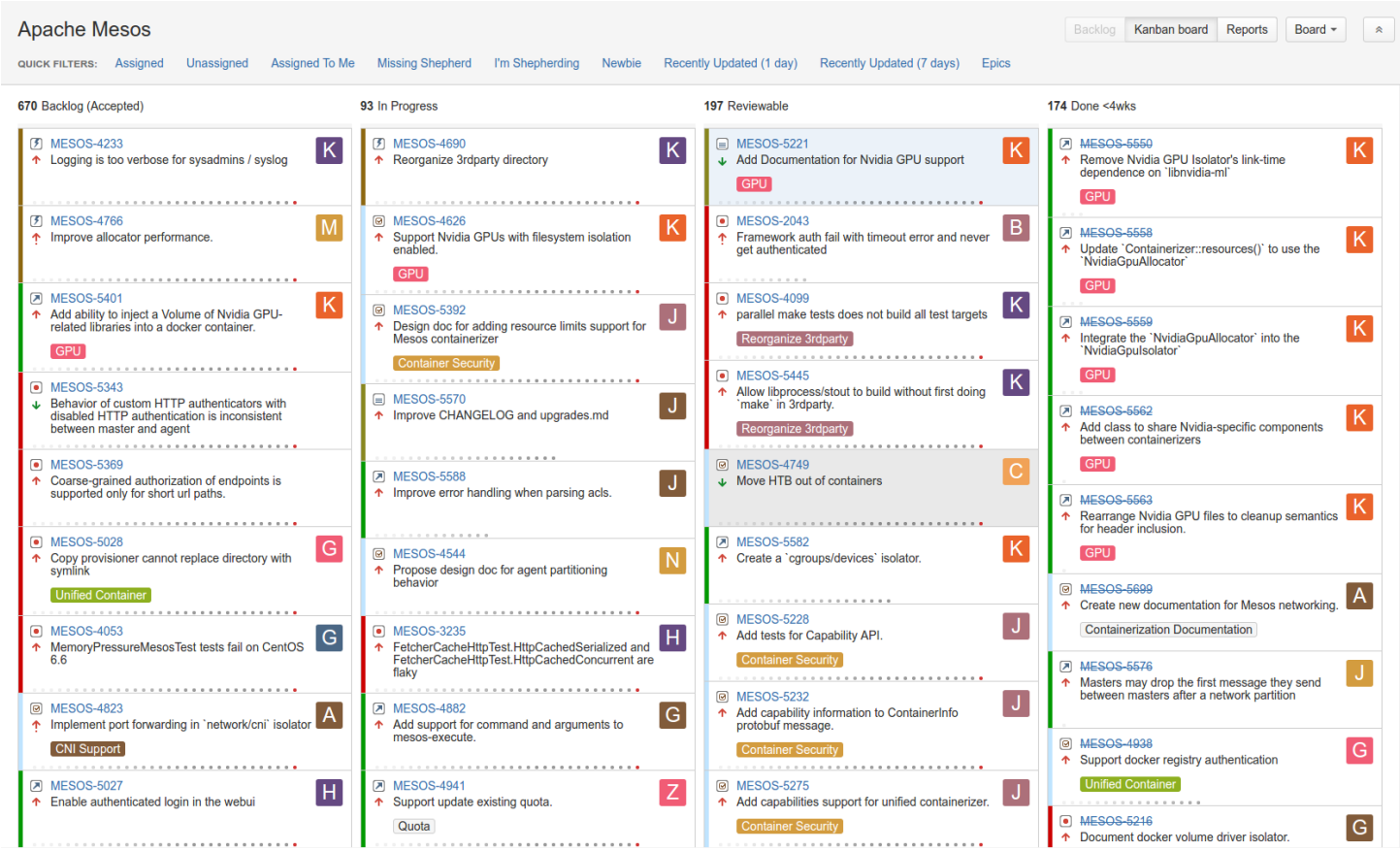


Figure 1 **Example of JIRA board with issues.**

The state of the art tool developed by *Danescu-Niculescu-Mizil et al.* (*2013*) was used to evaluate politeness within comment-messages. The authors proposed a machine learning approach for evaluating the politeness of a request posted in two different web applications: Wikipedia (https://en.wikipedia.org/wiki/Main_Page) and Stack Overflow (http://stackoverflow.com). Stack Overflow is well known in the software engineering field and is largely used by software practitioners; hence, the model that authors used in *Danescu-Niculescu-Mizil et al.* (*2013*) was suitable for our domain based on Jira issues, where developers post and discuss about technical aspects of issues. The authors provide a Web application (http://www.mpi-sws.org/~cristian/Politeness.html) and a library version of their tool. To prepare the training set for the machine learning approach, over 10,000 utterances were labeled using Amazon Mechanical Turk. The authors decided to restrict the residence of the annotators to the US and conducted a linguistic background questionnaire.

Since "politeness is a culturally defined phenomenon and what is considered polite in one culture can sometimes be quite rude or simply eccentric in another cultural context" (http://en.wikipedia.org/wiki/Politeness), the choice of limiting the residence of the annotators to the US could be interpreted as a weakness of the the tool. However, the annotators analysed comments written by authors from around the world and not only

from the US. Therefore, the possible bias introduced by annotators with a similar cultural background, is reduced and the different cultures of the developers involved in the analysis considered. The use of the tool would have been problematic, if annotators were from the US and they had analysed only comments written by authors from the US.

We considered 22 open source projects from one of the largest datasets of issues reports openly available (*Ortu et al.*, *2015d*). This paper aims to answer the following research questions:

- **Does a relationship exist between politeness and issues fixing time?**
  Issue fixing time for polite issues is shorter than issue fixing time for impolite and mixed issues.
- **Does politeness among developers affect the attractiveness of a project?**
  Magnetism and Stickiness are positively correlated with the percentage of polite comments.
- **Does the percentage of polite comments vary over time?**
  The percentage of polite comments does vary over time and in some cases it changes from lower percentage of polite comments to higher percentage of polite comments from two consecutive observation intervals. The percentage of polite comments over time is (for the majority of the projects in our corpus) seasonal and not random.
- **How does politeness vary with respect to JIRA maintenance types and issue priorities?**
  Comments related to issues with maintenance Bug, priority Minor and Trivial, tend to have a higher percentage of impolite comments. Issues with maintenance New Feature, priority Blocker and Critical tend to have a higher percentage of polite comments.

This paper is an extended version of earlier work by the same authors (*Ortu et al.*, *2015b*). We added eight new systems to the original corpus analysed in (*Ortu et al.*, *2015b*) and two new research question (RQ3 and RQ4), we also reviewed the RQ2 performing deeper statistical analysis. The remainder of this paper is structured as follows: In the next section, we provide related work. 'Experimental Setup' describes the dataset used for this study and our approach/rationale to evaluate the politeness of comments posted by developers. In 'Results,' we present the results and elaborate on the research questions we address. In 'Discussion' we present a discussion on the obtained results and 'Threats to Validity' discusses the threats to validity. Finally, we summarise the study findings and present plans for future work in 'Conclusions and Future Work.'

RELATED WORK

A growing body of literature has investigated the importance and the influence of human and social aspects, emotions and mood both in software engineering and software development. Research has focused on understanding how the human aspects of a technical discipline can affect final results (*Brief & Weiss*, *2002*; *Capretz*, *2003*; *Cockburn & Highsmith*, *2001*; *Erez & Isen*, *2002*; *Kaluzniacky*, *2004*), and the effect of politeness (*Novielli, Calefato & Lanubile*, *2014*; *Tan & Howard-Jones*, *2014*; *Winschiers & Paterson*, *2004*; *Tsay, Dabbish & Herbsleb*, *2014*; *Rousinopoulos, Robles & González-Barahona*, *2014*).

Feldt et al. (*2008*) focused on personality as a relevant psychometric factor and presented results from an empirical study about correlations between personality and attitudes to software engineering processes and tools. The authors found that higher levels of the personality dimension ''conscientiousness'' correlated with attitudes towards work style, openness to changes and task preference.

IT companies are also becaming more conscious of social aspects. *Ehlers* (*2015*) evaluated the efforts of IT companies in acquiring software engineers by emphasizing socialness in their job advertising. The research analyzed 75,000 jobs advertising from the recruiting platform Indeed and about 2,800 job ads from StackoverflowCareers to investigate correlations between social factors and the employee satisfaction of a work place. The findings showed that many companies advertise socialness explicitly. The Manifesto for Agile Development indicates that people and communications are more essential than procedures and tools (*Beck et al.*, *2001*).

Studies have also investigated the relationship between affect and work-related achievements, including performance (*Miner & Glomb*, *2010*) and problem-solving processes, such as creativity, (*Amabile et al.*, *2005*). Furthermore, strong evidence for emotional contagion on all its possible polarities has been found in a recent very large scale study (*Kramer, Guillory & Hancock*, *2014*). Therefore, affect is an interesting avenue for research in software engineering.

Steinmacher et al. (*2015*) analyzed social barriers that obstructed first contributions of newcomers (new developers joining an open-source project). The study indicated how impolite answers were considered as a barrier by newcomers. These barriers were identified through a systematic literature review, responses collected from open source project contributors and students contributing to open source projects.

Roberts, Hann & Slaughter (*2006*) conducted a study which revealed how the different motivations of open-source developers were interrelated, how these motivations influenced participation and how past performance influenced subsequent motivations.

Guzman & Bruegge (*2013*) and *Guzman* (*2013a*) have proposed prototypes and initial descriptive studies towards the visualization of affect over a software development process. In their work, the authors applied sentiment analysis to data coming from mailing lists, web pages, and other text-based documents of software projects. Guzman et al. built a prototype to display a visualization of the affect of a development team, and they interviewed project members to validate the usefulness of their approach. In another study, *Guzman, Azócar & Li* (*2014*), performed sentiment analysis of Github's commit comments to investigate how emotions were related to a project's programming language, the commits' day of the week and time, and the approval of the projects. The analysis was performed over 29 top-starred Github repositories implemented in 14 different programming languages. The results showed Java to be the programming language most associated with negative affect. No correlation was found between the number of Github stars and the affect of the commit messages.

Panichella et al. (*2015*) presented a taxonomy to classify app reviews into categories relevant to software maintenance and evolution, as well as an approach that merges three techniques (Natural Language Processing, Text Analysis, Sentiment Analysis) to

automatically classify app reviews into the proposed categories. The authors showed that the combined use of these techniques achieves better results (a precision of 75% and a recall of 74%) than results obtained using each technique individually (precision of 70% and a recall of 67%).

Pletea, Vasilescu & Serebrenik (*2014*) studied security-related discussions on GitHub, as mined from discussions around commits and pull requests. The authors found that security-related discussions account for approximately 10% of all discussions on GitHub and that more negative emotions were expressed in security-related discussions than in other discussions. These findings confirmed the importance of properly training developers to address security concerns in their applications as well as the need to test applications thoroughly for security vulnerabilities in order to reduce frustration and improve overall project atmosphere.

Garcia, Zanetti & Schweitzer (*2013*) analyzed the relation between the emotions and the activity of contributors in the Open Source Software project Gentoo. The case study built on extensive data sets from the project's bug tracking platform Bugzilla, to quantify the activity of contributors, and its mail archives, to quantify the emotions of contributors by means of sentiment analysis. The Gentoo project is known for a period of centralization within its bug triaging community. This was followed by considerable changes in community organization and performance after the sudden retirement of the central contributor. The authors analyzed how this event correlated with the negative emotions, both in bilateral email discussions with the central contributor, and at the level of the whole community of contributors. The authors also extended the study to consider the activity patterns of Gentoo contributors in general. They found that contributors were more likely to become inactive when they expressed strong positive or negative emotions in the bug tracker, or when they deviated from the expected value of emotions in the mailing list. The authors used these insights to develop a Bayesian classifier which detected the risk of contributors leaving the project.

Graziotin, Wang & Abrahamsson (*2015*) conducted a qualitative interpretive study based on face-to-face open-ended interviews, in-field observations and e-mail exchanges. This enabled the authors to construct a novel explanatory theory of the impact of affects on development performance. The theory was explicated using an established taxonomy framework. The proposed theory built upon the concepts of events, affects, attractors, focus, goals, and performance.

In other work *Graziotin, Wang & Abrahamsson* (*2014*) reported the results of an investigation with 42 participants about the relationship between the affective states, creativity, and analytical problem-solving skills of software developers. The results offered support for the claim that happy developers were better problem solvers in terms of their analytical abilities. The authors provided a better understanding of the impact of affective states on the creativity and analytical problem-solving capacities of developers, introduced and validated psychological measurements, theories, and concepts of affective states, creativity, and analytical-problem-solving skills in empirical software engineering, and raised the need for studying the human factors of software engineering by employing a multi-disciplinary viewpoint.

Rigby & Hassan (*2007*) analyzed, using a psychometrically-based linguistic analysis tool, the five big personality traits of software developers in the Apache httpd server mailing list. The authors found that two developers responsible for the major Apache releases had similar personalities and their personalities were different from other developers. *Bazelli, Hindle & Stroulia* (*2013*) analyzed questions and answers on stackoverflow.com to determine the developer personality traits, using the Linguistic Inquiry and Word Count (*Pennebaker, Francis & Booth*, *2001*). The authors found that the top reputed authors were more extroverted and expressed less negative emotions than authors of down voted posts.

Tourani, Jiang & Adams (*2014*) evaluated the use of automatic sentiment analysis to identify distress or happiness in a team of developers. They extracted sentiment values from the mailing lists of two mature projects of the Apache software foundation, considering developers and users. The authors found that an automatic sentiment analysis tool obtained low precision on email messages (due to long size of the analyzed text) and that users and developers express positive and negative sentiment on mailing lists.

Murgia et al. (*2014b*) analyzed whether issue reports carried any emotional information about software development. The authors found that issue reports contain emotions regarding design choices, maintenance activity or colleagues. *Gómez et al.* (*2012*) performed an experiment to evaluate whether the level of extraversion in a team influenced the final quality of the software products obtained and the satisfaction perceived while this work was being carried out. Results indicated that when forming work teams, project managers should carry out a personality test in order to balance the amount of extraverted team members with those who are not extraverted. This would permit the team members to feel satisfied with the work carried out by the team without reducing the quality of the software products developed.

Acuña, Gómez & Juristo (*2008*), performed empirical research examining the work climate within software development teams. The authors attempted to understand if team climate (defined as the shared perceptions of team work procedures and practices) bore any relation to software product quality. They found that high team vision preferences and high participative safety perceptions of the team were significantly related to better software. In a study conducted by *Fagerholm et al.* (*2014*), it was shown that software teams engaged in a constant cycle of interpreting their performance. Thus, enhancing performance experiences requires integration of communication, team spirit and team identity into the development process.

Jongeling, Datta & Serebrenik (*2015*) studied whether the sentiment analysis tools agreed with the sentiment recognized by human evaluators as well as with each other. Furthermore, the authors evaluated the impact of the choice of a sentiment analysis tool on software engineering studies by conducting a simple study of differences in issue resolution times for positive, negative and neutral texts. The authors repeated the study for seven datasets and different sentiment analysis tools and observed that the disagreement between the tools can lead to contradictory conclusions.

Table 1 Selected Projects Statistics.

Project	# of comments	# of developers
HBase	91,016	951
Hadoop Common	61,958	1,243
Derby	52,668	675
Lucene Core	50,152	1,107
Hadoop HDFS	42,208	757
Cassandra	41,966	1,177
Solr	41,695	1,590
Hive	39,002	850
Hadoop Map/Reduce	34,793	875
Harmony	28,619	316
OFBiz	25,694	578
Infrastructure	25,439	1,362
Camel	24,109	908
ZooKeeper	16,672	495
GeoServer	17,424	705
Geronimo	18,017	499
Groovy	18,186	1,305
Hibernate ORM	23,575	4,037
JBoss	23,035	453
JRuby	22,233	1,523
Pig	21,662	549
Wicket	17,449	1,243
Tot	737,572	18,144

EXPERIMENTAL SETUP

Dataset

We built our dataset collecting data from the Apache Software Foundation Issue Tracking system, JIRA (https://www.atlassian.com/software/jira). An Issue Tracking System (ITS) is a repository used by software developers to support the software development process. It supports corrective maintenance activity like Bug Tracking systems, along with other types of maintenance requests. We mined the ITS of the Apache Software Foundation collecting issues from October 2002 to December 2013. In order to create our dataset, since the focus of our study was about the usefulness of Agile boards, we selected projects for which the JIRA Agile board contained a significant amount of activity. Namely, we selected those systems for which the Agile board contained more than 15,000 comments (in order to build a time series with sufficient data) and there was a recorded monthly activity (i.e., developers were active for every considered month). The JIRA dataset contains roughly 1,000 projects, 150 of which characterised by more than 1,000 comments. Table 1 shows the corpus of 22 projects selected for our analysis, highlighting the number of comments recorded for each project and the number of developers involved.

Table 2 Examples of polite and impolite comments.

Comment	POLITE
Hey <dev_name_a>, Would you be interested in contributing a fix and a test case for this as well? Thanks, <dev_name_b>	YES
<dev_name>, can you open a new JIRA for those suggestions? I'll be happy to review.	YES
<dev_name>, the latest patch isn't applying cleanly to trunk - could you resubmit it please? Thanks.	YES
<dev_name>, Since you can reproduce, do you still want the logs? I think I still have them if needed.	YES
Why are you cloning tickets? Don't do that.	NO
shouldnt it check for existence of tarball even before it tries to allocate and error out ???	NO
<dev_name_a>, why no unit test? <dev_name_b>, why didn't you wait for +1 from Hudson???	NO
> this isn't the forum to clarify Why not? The question is whether this is redundant with Cascading, so comparisons are certainly relevant, no?	NO

Comments politeness

Given some texts, the tool developed by *Danescu-Niculescu-Mizil et al.* (*2013*) calculates the politeness of its sentences providing one of two possible labels: *polite* or *impolite* as result. *impolite*. Table 2 shows some examples of polite and impolite comments as classified by the tool.[1]

[1] User's names are reported as *<dev_name_a>* for the sake of privacy.

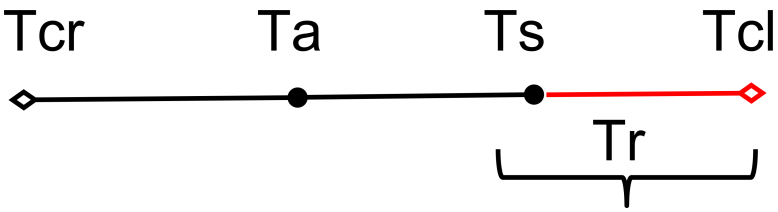


Figure 2 **Example of timeline for JIRA issue.**

We evaluated the percentage of polite comments *per* month considering all comments posted in a certain month. For each comment we assigned a value according to the following rules:

- Value of $+1$ for those comments marked as polite by the tool.
- Value of -1 for those comments marked as impolite.

Finally, we calculated the percentage of polite comments for a certain month. We analyzed the politeness of about 700K comments.

Issue politeness

The next step was to infer the politeness of issues from the knowledge of comments politeness. We grouped issues together as follows:

- we first divided comments in polite sets: polite, impolite;
- we divided issues in three sets: polite issues (commented only with polite comments), impolite issues (commented only by impolite comments) and mixed issue (commented with both polite and impolite comments).

For each issue we evaluated the politeness expressed in its comments and we then divided issues in three groups: polite issues containing polite comments, impolite issues containing impolite comments and mixed issues containing both polite and impolite comments. Our dataset contains in total 174,871 issues, 5.3% (9,269) of the total were classified as polite, 56.9% (99,501) classified as impolite, 37.8% (66,101) classified as mixed. For each of this three groups of issues we evaluated the issue fixing time as the difference between resolution and creation time. Figure 2 shows the typical issue timeline in JIRA:

- T_{cr} represents the time when an issue is created.
- T_{cl} represents the time when an issue is closed.
- T_{a} represents the time when an issue is assigned to a developer.
- T_{s} is the time when a developer subscribes to an issue that has been assigned to them.

To infer the issue fixing time (abbreviated as IFT), we used the approach proposed by *Murgia et al.* (*2014a*). We computed the time interval between the last time an issue had been closed and the last time it had been subscribed to by an assignee.

Attractiveness

Our research focuses around the concepts developed by *Yamashita et al.* (*2014*) and *Yamashita et al.* (*2016*) who introduced the concepts of magnetism and stickiness for

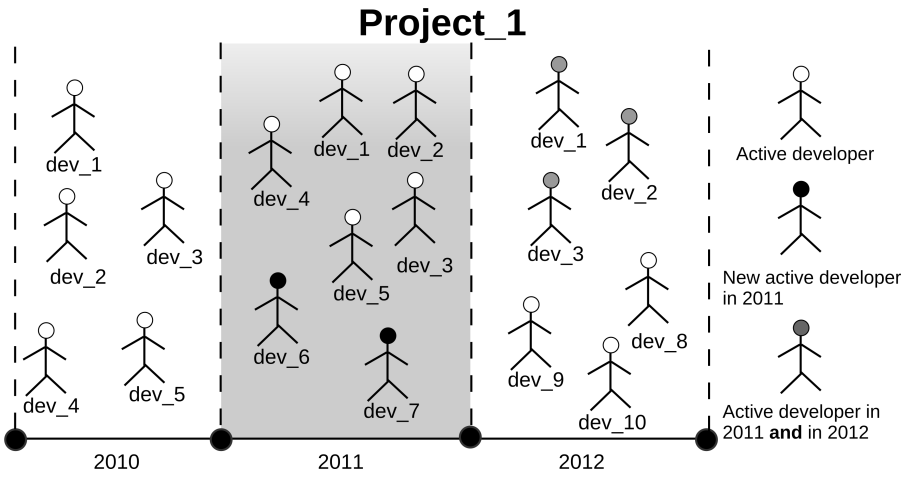


Figure 3 **Example of Magnetism and Stickiness metrics computation in 2011.**

a software project. A project is classified as *Magnetic* if it has the ability to attract new developers over time. *Stickiness* is the ability of a project to keep its developers over time. We measured these two metrics by considering the period of observation of one year. Figure 3 shows an example of the evaluation of Magnetism and Stickiness metrics. In this example, we were interested in calculating the value of Magnetism and Stickiness for 2011. From 2010 to 2012, we had a total of 10 *active*[2] developers. In 2011, there were seven active developers and 2 of them (highlighted with black heads) were new. Only 3 (highlighted with grey heads) of the seven active developers in 2011 were also active in 2012. We can then calculate the Magnetism and Stickiness as follows:

- *Magnetism* is the fraction of new active developers during the observed time interval, in our example 2/10 (*dev_6* and *dev_7* were active in 2011 but not in 2010).
- *Stickiness* is the fraction of active developers that were also active during next time interval, in our example 3/7 (*dev_1*, *dev_2*, *dev_3* were active in 2011 and in 2012).

[2]We consider as active all developers who posted/commented/resolved/modified an issue during the observed time (from dev_1 to dev_10).

Data analysis

To perform statistical testing, filter the data and produce the visualisation of the results, we used *scikit-learn* (http://scikit-learn.org/stable/) for RQ1, and the R projects for statistical computing (*R Development Core Team*, *2014*) for the other RQs. To facilitate replication of our study, we have created a replication package (https://bitbucket.org/giuseppedestefanis/peerjcs_replicationpackage) which contains the dataset, the tool used to detect politeness and the R and Python scripts for performing the statistical analysis.

RESULTS

Does a relationship exist between politeness and issues fixing time?

Motivation. *Murgia et al.* (*2014a*) demonstrated the influence of maintenance type on the issue fixing time, while *Zhang, Gong & Versteeg* (*2013*) developed a prediction model for bug fixing time for commercial software.

In another study, *Zhang et al.* (*2012*) analyzed the interval between bug assignment and the time when bug fixing starts (bugs are classified as a type of issue in JIRA). After a bug being reported and assigned, some developers will immediately start fixing the bug while others will start bug fixing after a long period. The authors explored the delays of developers empirically analyzing three open source software systems. They studied factors affecting bug fixing time along three dimensions (bug reports, source code involved in the fix and code changes that are required to fix the bug) and compared such factors using logistic regression models. The most influential factors on bug fixing appeared to be the severity of a bug, its description and comments and operating system where a bug was found. Hence, there are indeed many factors able to influence bugs fixing time and issues fixing time; in this case, we were interested in finding out if politeness expressed by developers in comments had an influence on issue fixing time.

To detect differences among the fixing time of polite, impolite and mixed issues, we used the Kruskal–Wallis test. Such a test is non parametric and unpaired (*Siegel*, *1956*; *Kruskal & Wallis*, *1952*; *Weiss et al.*, *2007*). The test can be used with no restrictions or hypotheses on the statistical distribution of the sample populations. The test is suitable for comparing differences among the medians of two or more populations when their distributions are not gaussian. We grouped all the issues (by category) of all the projects contained in our corpus and then we tested the null hypothesis H_0: *the three distributions of issue fixing time are equal for the three typologies of considered issues (polite, impolite, mixed)*. The outcome of the Kruskal–Wallis test is a *p-value* $p < 2^{-16}$ indicating that the three distributions are statistically different. Figure 4 shows the box-plot of the issues fixing time for the three groups of issues considered (polite, impolite and mixed) for all the issues analysed. The issues fixing time is expressed in hours on a logarithmic scale. The median of the issue fixing time for polite issues is shorter than that for impolite and mixed issues, while the median for impolite issues is shorter than the one for mixed issues. Figure 4 also shows that the percentage of impolite issues is the highest (56.9%), followed by mixed issues (37.7%) and then polite issue (5.3%).

Figures 5 and 6 show the boxplots of the issue fixing time when considering single projects. We considered the four projects (Hadoop HDFS, Derby, Lucene-Core, Hadoop Map/Reduce) with the highest number of comments in our corpus as examples. For visualising the boxplots of Figs. 5 and 6, we grouped the issues by project and then by category. The results obtained with all the issues grouped together (without distinguishing for project) are confirmed also with these four examples. The median of the issues for polite issues is lower than the other two medians, and also in this cases issues with mixed polite and impolite comments have a longer issue fixing time than issues with impolite comments.

Findings. *Issue fixing time for polite issues is shorter than issue fixing time for impolite and mixed issues.*

Does politeness among developers affect the attractiveness of a project?

Motivation. Magnetism and Stickiness are two interesting metrics able to describe the general health of a project; namely, if a project is able to attract new developers and to

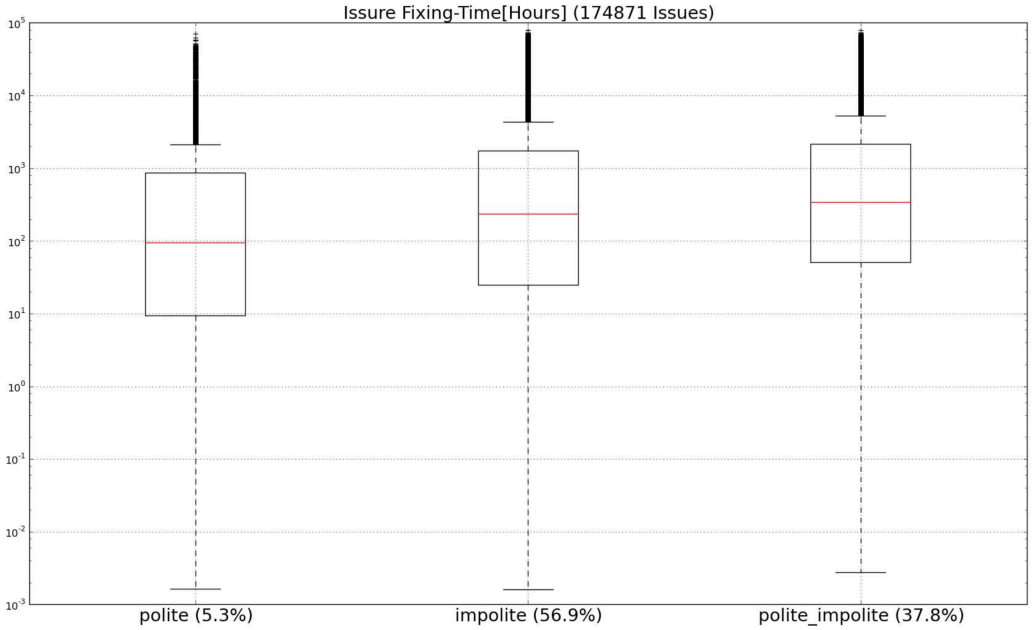
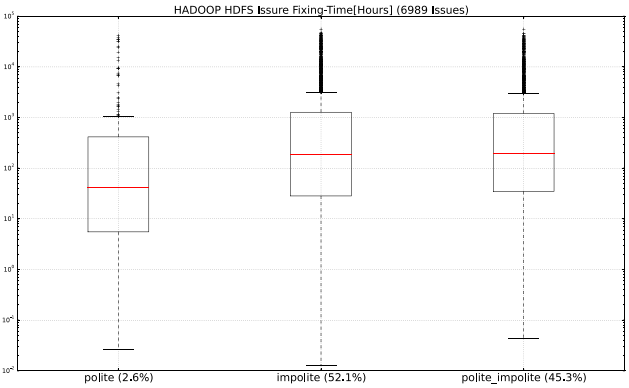
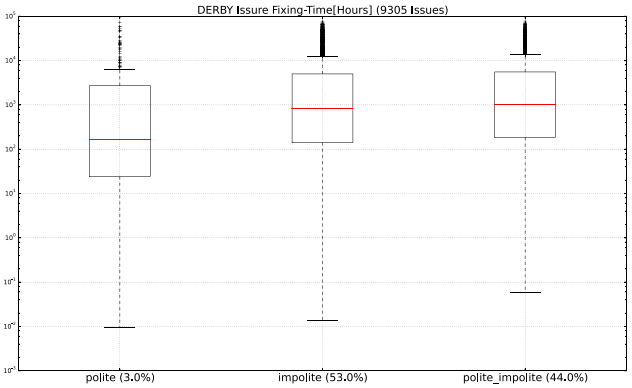


Figure 4 **Box-plot of the fixing-time expressed in Hours.** The number in parentheses next to issue group indicates the percentage of issues.



(a) Hadoop HDFS

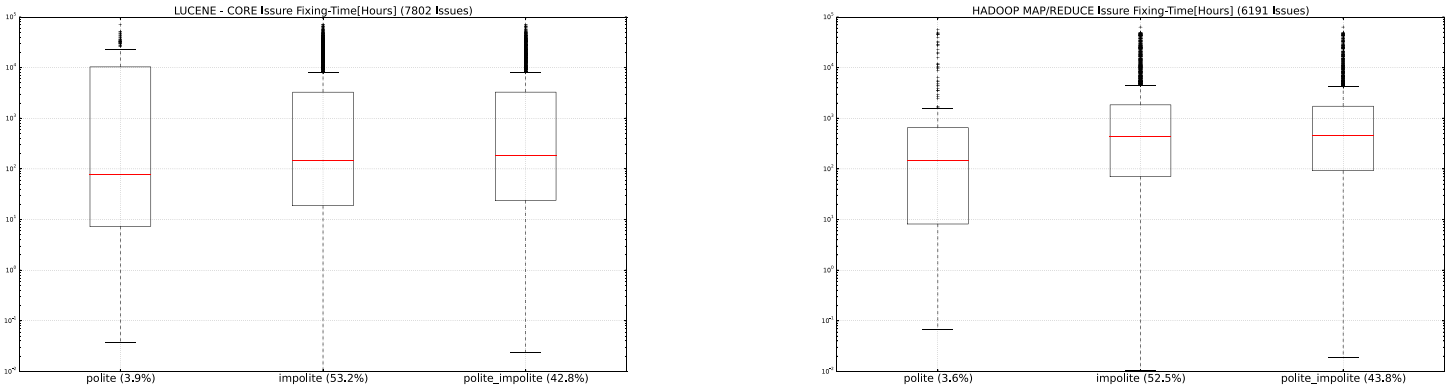


(b) Derby

Figure 5 **Box-plot of the fixing-time expressed in Hours for single projects.** The number in parentheses next to polite/impolite indicates the percentage of impolite and polite issues.

keep them over time we can then conclude that the project is healthy. On the other hand, if a project is not magnetic and is not sticky we can conclude that the project is losing developers and is not attracting new developers over time. Although there may be many factors influencing magnetism and stickiness, we were interested in analysing the correlation between politeness expressed by developers in their comments and these two metrics.

To detect if there was a direct correlation between magnetism and stickiness of a project and politeness, we considered an observation time of one month. During this time interval



(a) Lucene-Core **(b)** Hadoop Map/Reduce

Figure 6 Box-plot of the fixing-time expressed in Hours for single projects. The number in parentheses next to polite/impolite indicates the percentage of impolite and polite issues.

we measured magnetism, stickiness and the percentage of polite comments. Since we had no evidence that the politeness in the observed time could affect magnetism and stickiness in the same time interval or in the following observation time, we evaluated a cross-correlation coefficient. To study the cross correlation, we first studied the time series for stationarity. A time series $X_t(t = 1, 2...)$ is considered to be stationary if its statistical properties (autocorrelation, variance, expectation) do not vary with time (*Priestley*, *1981*; *Priestley*, *1988*; *Hamilton*, *1994*). Time series stationarity is a condition required for calculating the most common correlation coefficient. However, it is still possible to study cross-correlation between time series which are not stationary (*Krištoufek*, *2014*).

We used the R package *fpp* (https://cran.r-project.org/package=fpp) and we applied:

- Ljung–Box test (*Box & Pierce*, *1970*; *Ljung & Box*, *1978*): this test for stationarity confirms independence of increments, where rejection of the null hypothesis H_0 indicates stationarity (the null hypothesis H_0 is that the data are non-stationary);
- Augmented Dickey-Fuller (ADF) t-statistic test (*Said & Dickey*, *1984*; *Diebold & Rudebusch*, *1991*; *Banerjee et al.*, *1993*): in the Augmented Dickey-Fuller (ADF) t-statistic test the null hypothesis H_0 is that the data are non-stationary (small p-values (e.g., less than 0.05) suggest that the time-series is stationary).
- Kwiatkowski-Phillips–Schmidt–Shin test (KPSS) (*Kwiatkowski et al.*, *1992*): this test reverses the hypotheses, hence the null-hypothesis H_0 is that the time-series is stationary. Small p-values (e.g., less than 0.05) suggest that the time-series is not stationary.

We decided to proceed using the results obtained from the three tests used for checking stationarity of a time series and then consider the worst case scenario (e.g., even if only one test out of three indicated rejection of the hypothesis of stationarity for a given time series, we considered that time series as non stationary). The results of the three tests are shown in Table 3. The cells in grey indicate that the p-value for the corresponding test is below 5% (our cutoff for significance), thus we infer in these cases that the test indicates stationarity

Table 3 *P*-value results for stationarity tests (Ljung–Box, Augmented Dickey-Fuller, KPSS).

Project	Politeness			Magnetism			Stickiness		
HBase	0.00023	0.028	0.01	0.9765	0.01	0.0285	0.266	0.156	0.1
Hadoop C.	0.00072	0.59	0.044	0.0677	0.01	0.1	3.062e-08	0.01	0.01
Derby	1.626e-08	0.01	0.01	0.19	0.049	0.1	0.037	0.094	0.01
Lucene C.	0.498	0.0173	0.017	0.38	0.08	0.01	3.105e-13	0.48	0.01
Hadoop HDFS	0.025	0.79	0.09	0.007	0.01	0.01	0.093	0.30	0.036
Cassandra	0.25	0.043	0.1	0.57	0.18	0.1	0.099	0.387	0.01
Solr	0.0075	0.31	0.01	0.0003	0.01	1.615e-07	0.01	0.01	0.01
Hive	8.075e-08	0.113	0.01	0.335	0.32	0.01	0.006	0.28	0.01
Hadoop MR	6.815e-05	0.4	0.01	0.0003	0.06	0.01	8.821e-08	0.92	0.012
Harmony	0.03	0.012	0.01	0.0019	0.23	0.058	3.658e-08	0.01	0.01
OFBiz	0.0006	0.01	0.01	0.91	0.09	0.1	7.986e-05	0.01	0.01
Infrastructure	0.16	0.01	0.1	0.0002	0.01	0.99	0.53	0.028	0.01
Camel	0.00097	0.01	0.01	0.79	0.01	0.1	4.885e-15	0.36	0.01
ZooKeeper	0.14	0.45	0.046	0.26	0.01	0.1	0.398	0.17	0.01
GeoServer	0.064	0.01	0.01	0.0005	0.01	0.1	0.49	0.027	0.011
Geronimo	0.32	0.01	0.1	0.84	0.078	0.1	2.665e-15	0.59	0.01
Groovy	8.514e-05	0.01	0.047	0.14	0.01	0.1	1.225e-07	0.01	0.01
Hibernate ORM	4.393e-05	0.063	0.016	0.15	0.01	0.06	3.074e-12	0.01	0.033
JBoss	0.36	0.01	0.1	0.53	0.95	0.027	2.2e-16	0.88	0.01
JRuby	0.67	0.29	0.02	0.074	0.01	0.1	0.00075	0.32	0.01
Pig	7.096e-05	0.12	0.02	0.59	0.01	0.078	2.2e-16	0.45	0.01
Wicket	0.084	0.064	0.1	0.093	0.202	0.1	1.105e-05	0.33	0.018

(on the contrary, cells in white indicate that the *p*-value for the corresponding test is above 5%). For example, for the percentage of polite comments time series of HBase (row 1, first three cells), we have that the first two tests (Box-Ljung and Augmented Dickey-Fuller) suggest stationarity, while the third test (KPSS) rejects the hypothesis of stationarity. For the majority of the cases, the tests provides discordant results. Only for Magnetism for Infrastructure (row 12) and GeoServer (row 15) there is agreement among the three tests. Thus, we considered all the time series in Table 3, except for the cases mentioned before, being not stationary. Table 4 shows the results obtained applying the algorithm illustrated in *Krištoufek* (*2014*) (http://stats.stackexchange.com/questions/149799/code-for-detrended-cross-correlation-in-r). For the majority of the cases there is a weak positive correlation between politeness and Magnetism (14 projects out 22) and politeness and Stickiness (13 projects out 22).

We also calculated the cross correlation (using the ccf function in R) after applying time series differencing to transform time series that were not stationary in Table 3 into stationary. The D differencing operator applied to a time series k is to create a new series k_n whose value at time t is the difference between $k(t+t_1)$ and $k(t)$. This method is useful for removing cycles and trends.

Table 4 Cross-Correlation for not stationary series.

Project	Politeness-Magnetism	Politeness-Stickiness
HBase	0.131	−0.096
Hadoop Common	0.024	0.018
Derby	−0.0018	0.0147
Lucene Core	0.32	0.137
Hadoop HDFS	−0.08	0.104
Cassandra	0.296	−0.136
Solr	−0.169	0.0018
Hive	0.305	0.184
Hadoop Map/Reduce	−0.047	0.052
Harmony	0.21	0.026
OFBiz	0.11	0.097
Infrastructure	−0.025	0.292
Camel	0.115	0.013
ZooKeeper	−0.19	−0.11
GeoServer	0.22	−0.13
Geronimo	0.23	0.27
Groovy	0.013	−0.22
Hibernate ORM	0.02	−0.1
JBoss	−0.26	−0.2
JRuby	−0.12	−0.06
Pig	0.26	0.14
Wicket	0.016	−0.14

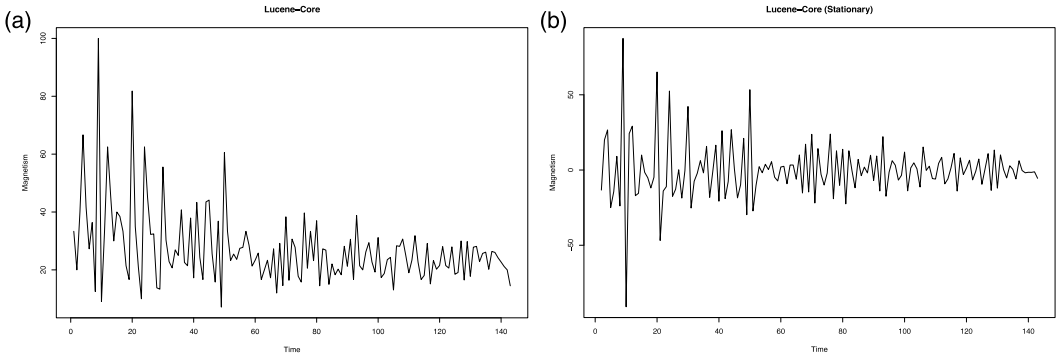


Figure 7 Differencing Time-series.

As an example, Fig. 7A shows the Magnetism time-series for Lucene Core. From Table 3, row 4, we can see that the time series is not stationary, since all the three test failed in proving stationarity. By applying the differencing operator (first differencing), we obtain the new time-series in Fig. 7B.

The new time-series in Fig. 7B is stationary; all the three tests provide the same indication for stationarity (Box-Ljung: $p - value = 1.4e - 13$, Augmented Dickey-Fuller:

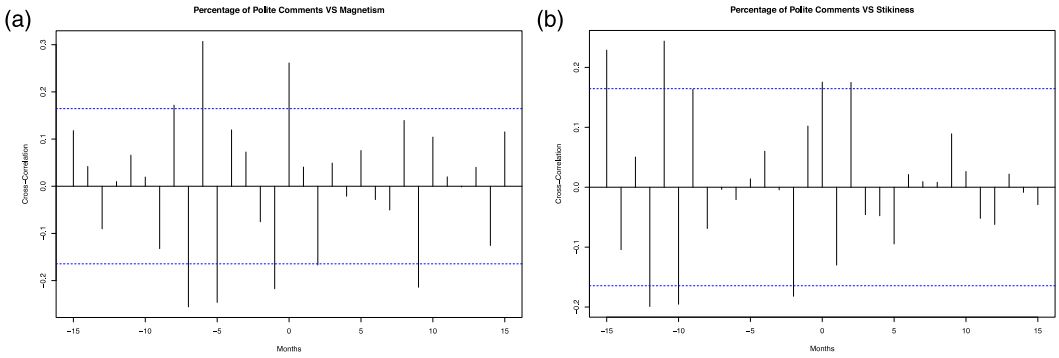


Figure 8 **Cross-Correlation—Lucene Core.**

$p-value = 0.01$, KPSS: $p-value = 0.1$) and the plot appears roughly horizontal (the visual inspection is a practical rule of thumb which can help when evaluating a time-series for stationarity). If the time-series under study are stationary, it is possible to calculate the cross correlation. The cross correlation function (ccf) is defined as the set of correlations (height of the vertical line segments in Fig. 8) between two time series $x_t + h$ and y_t for lags $h = 0, \pm 1, \pm 2, \ldots$. A negative value for h represents a correlation between the x-series at a time before t and the y-series at time t. For example, if the lag $h = -1$, then the cross correlation value would give the correlation between $x_t - 1$ and y_t. On the contrary, negative lines correspond to anti-correlated events.

The *ccf* helps to identify lags of x_t that could be predictors of the y_t series.

- When $h < 0$ (left side of plots in Figs. 8A and 8B), x leads y.
- When $h > 0$ (right side of plots in Figs. 8A and 8B), y leads x.

Table 5 shows the maximum value of the cross-correlation coefficient between the percentage of polite comments and Magnetism and the percentage of polite comments and Stickiness and the lag (or lags) in which the maximum value occurs. The values are calculated using the R function *ccf*.

A negative lag x means that the current values of Stickiness are likely to be higher, if x month before the percentage of polite comments was higher. On the other hand, a positive lag z means that a current higher percentage of polite comments is linked with higher Magnetism z months later. For both Magnetism and Stickiness, we observe that a positive maximum correlation exists.

The difference sin lags presented in Table 5 could be explained looking at the composition of our corpus. We selected the projects with a higher number of comments from JIRA, regardless of domain, history and/or programming language used. Additionally, there are systems which are younger than others and, as a consequence, the time series may have different lengths.

Findings. *Magnetism and Stickiness are positively correlated with the percentage of polite comments.*

Table 5 Politeness vs Magnet and Sticky Cross-Correlation Coefficient.

Project	Pol Vs Mag	Lag	Pol Vs Stick	Lag
HBase	0.378	−1,7	0.418	6
Hadoop Common	0.283	−5	0.203	5
Derby	0.181	15	0.185	−1
Lucene Core	0.31	−6	0.243	−11
Hadoop HDFS	0.3	0	0.32	−1
Cassandra	0.414	5	0.315	2
Solr	0.33	−4	0.23	12
Hive	0.4	−14	0.3	−13
Hadoop Map/Reduce	0.35	3	0.35	3
Harmony	0.3	−5	0.35	1
OFBiz	0.25	14	0.22	−11
Infrastructure	0.17	4	0.26	−11
Camel	0.28	−12	0.26	−14,-2,7
ZooKeeper	0.41	6	0.27	−2
GeoServer	0.3	0	0.2	−5,−1
Geronimo	0.32	0	0.24	0
Groovy	0.2	3	0.21	2
JBoss	0.3	8	0.43	5
Hibernate ORM	0.23	1	0.18	−3
JRuby	0.25	10	0.2	−10
Pig	0.38	11	0.17	7,9
Wicket	0.3	−13	0.27	11

Does the percentage of polite comments vary over time?

Motivation. Politeness has an influence on the productivity of a team (*Ortu et al.*, *2015b*; *Ortu et al.*, *2015a*; *Ortu et al.*, *2015c*). Thus, it is interesting to understand if there are periods of time in which the level of politeness decreases (potentially affecting the productivity of a team).

We calculated the level of politeness for any given issue and then plotted the percentage of polite comments *per* month grouping issues *per* project. For each project considered in this study, the percentage of polite comments over time can be seen as time series, hence we performed tests for randomness and seasonality to understand the nature of politeness time series. A time series is considered random if it consists of independent values from the same distribution. We used the Bartels test (*Bartels*, *1982*) for studying randomness (*Brockwell & Davis*, *2006*) and the results from the Augmented Dickey-Fuller test and KPSS test from 'Does politeness among developers affect the attractiveness of a project?' for studying seasonality. We used the R package *randtest* (https://cran.r-project.org/web/packages/randtests/randtests.pdf) and we applied:

- Bartels test: in this test, the null hypothesis H_0 of randomness is tested against non randomness.

Table 6 Randomness and seasonality test results for Politeness.

	Randomness	*Seasonality*	
Project	**Bartels-rank *p*-value**	**Aug. D-F *p*-value**	**KPSS**
HBase	0.0006529	0.028	0.01
Hadoop Common	3.362e-06	0.59	0.044
Derby	6.824e-08	0.01	0.01
Lucene Core	0.008213	0.0173	0.017
Hadoop HDFS	1.622e-06	0.79	0.09
Cassandra	0.3747	0.043	0.1
Solr	0.0026	0.31	0.01
Hive	1.044e-09	0.113	0.01
Hadoop Map/Reduce	0.0004533	0.4	0.01
Harmony	0.001998	0.012	0.01
OFBiz	0.0003783	0.01	0.01
Infrastructure	0.02888	0.01	0.1
Camel	2.951e-06	0.01	0.01
ZooKeeper	0.07181	0.45	0.046
GeoServer	0.0008848	0.01	0.01
Geronimo	0.0702	0.01	0.1
Groovy	0.00025	0.01	0.047
Hibernate ORM	0.000384	0.063	0.016
JBoss	0.2112	0.01	0.1
JRuby	0.6816	0.29	0.02
Pig	1.957e-05	0.12	0.02
Wicket	0.1448	0.064	0.1

For studying the seasonality of the percentage of polite comments time series, we considered the results (from 'Does politeness among developers affect the attractiveness of a project?') of the following tests:

- Augmented Dickey Fuller test: the null hypothesis H_0 is that the data are non-stationary and **non-seasonal**;
- Kwiatkowski-Phillips–Schmidt–Shin (KPSS) test: the null hypothesis H_0 is that the data are stationary and **non-seasonal**.

The results of the tests are shown in Table 6.

For randomness, the cells in grey indicate that the *p*-value for the corresponding test is higher than 5% (our cutoff for significance), thus we infer in these cases that the test indicates randomness (null hypothesis H_0 of randomness). On the contrary, cells in white indicate that the *p*-value for the corresponding test is lower than 5%. In the majority of the cases (16 out 22), the percentage of polite comments time series were not random. For seasonality, the cells in grey indicate that the *p*-value for the corresponding test is less than 5% (our cutoff for significance), thus we infer in these cases that the test rejects the null hypothesis H_0 of **non-seasonality**.

Table 6 shows that for the Augmented Dickey Fuller test, 12 projects (HBase, Derby, Lucene Core, Cassandra, Harmony, OFBiz, Infrastructure, Camel, GeoServer, Geronimo, Groovy, JBoss) have a percentage of polite comments time series which presents seasonality, while 10 time series are not seasonal. For the KPSS test, the null hypothesis of non-seasonality is rejected for 16 time series (out of 22).

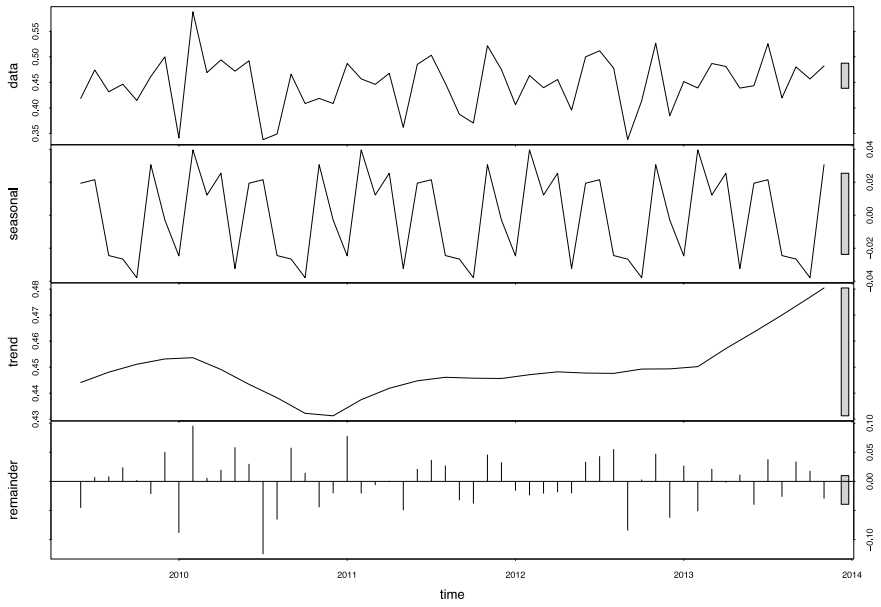
It is interesting to note that there are variations in the percentage of polite comments over time. This is by no means a representation of a time dynamics, but simply the representation of random variation of the percentage of polite comments over time. Cassandra and OFBiz present seasonality (Table 6) and Fig. 9 shows the seasonal component determined using the *stl* function (https://stat.ethz.ch/R-manual/R-devel/library/stats/html/stl.html) in R. In Cassandra, Fig. 9(A), and OFbiz, Fig. 9(B), we see how the percentage of polite comments decreases for some time interval and increases for some others. It is also possible to analyse the percentage of polite comments trend, while for Cassandra the trend is increasing starting from the year 2011, for OFBiz the trend is decreasing.

Findings. *The percentage of polite comments does vary over time and in some cases it changes from lower percentage of polite comments to higher percentage of polite comments from two consecutive observation intervals.* This fact could be related to the composition of our corpus. We considered only open source systems, hence there are no strict deadlines or particular busy days (such as Fridays, as suggested by *Śliwerski, Zimmermann & Zeller* (*2005*)).

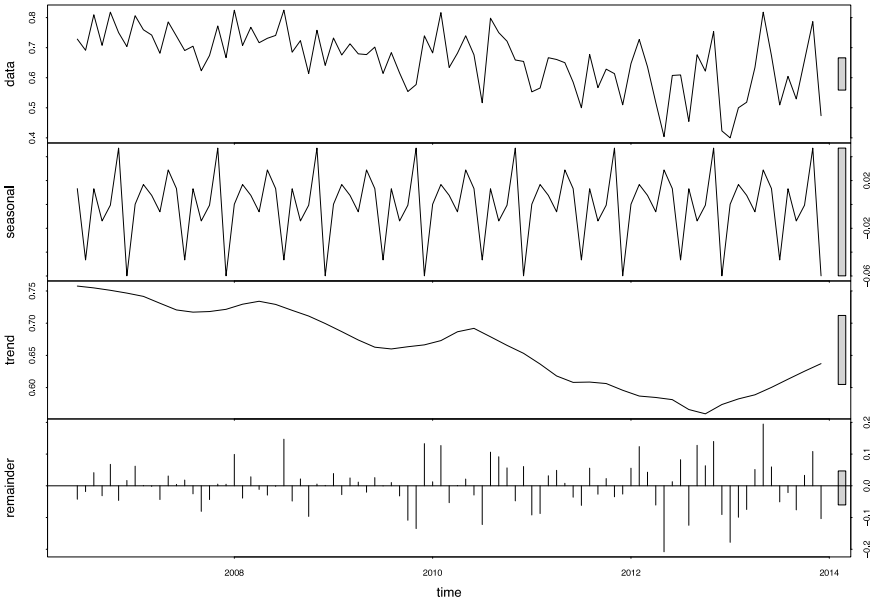
How does politeness vary with respect to JIRA maintenance types and issue priorities?

Motivation. Understanding which typology of issue attracts more impolite comments could help both managers and developers better understand the development process and take action to better manage the distribution of issues within development teams. A classification of the type of issues, is provided on the JIRA wiki (https://cwiki.apache.org/confluence/display/FLUME/Classification+of+JIRA+Issues). The following list gives a brief introduction:

- Bug: this type of issue indicates a defect in the source code, such as logic errors, out-of-memory errors, memory leaks and run-time errors. Any failure of the product to perform as expected and any other unexpected or unwanted behaviour can be registered as type Bug.
- SubTask: this type of issue indicates that a task must be completed as an element of a larger and more complex task. Subtask issues are useful for dividing a parent issue into a number of smaller tasks, more manageable units that can be assigned and tracked separately.
- Task: this type of issue indicates a task that it is compulsory to complete.
- Improvement: this type of issue indicates an improvement or enhancement to an existing feature of the system.
- New Feature: this type of issue indicates a new feature of the product yet to be developed.



(a) Cassandra



(b) OfBiz

Figure 9 **Time series decomposition.**

Table 7 Maintenance statistics.

Type	IMPOLITE	POLITE
Bug	201,359	163,489
Sub-task	24,333	22,909
Task	19,379	14,442
Improvement	90,477	90,564
New Feature	33,640	39,538
Wish	2573	2681
Test	3788	3270
New JIRA Project	172	210
Brainstorming	98	172
Umbrella	87	144

Table 8 Priority statistics.

Priotity	IMPOLITE	POLITE
Blocker	20,657	19,049
Critical	21,517	19,410
Major	241,012	219,841
Minor	82,892	71,905
Optional	105	52
Trivial	12,009	8,479

- Wish: this type of issue is used to track general wishlist items, which could be classified as new features or improvements for the system under development.
- Test: this type of issue can be used to track a new unit or integration test.
- New JIRA Project: this type of issue indicates the request for a new JIRA project to be set up.
- Brainstorming: this type of issue is more suitable for items in their early stage of formation not yet mature enough to be labelled as a Task or New Feature. It provides a bucket where thoughts and ideas from interested parties can be recorded as the discussion and exchange of ideas progresses. Once a resolution is made, a Task can be created with all the details defined during the brainstorming phase.
- Umbrella: this type of issue is an overarching type comprised of one or more sub-tasks.

Tables 7 and 8 provide information about the absolute number of issues (maintenance e priority) in our corpus.

To detect the level of politeness for each category of issue, we grouped the issue comments for type of maintenance and priority.

To justify claims such as ‘‘issues of type A tend to have more polite comments than issues of type B’’, we used the multiple contrast test procedure (*Konietschke, Hothorn & Brunner*, *2012*) using a 5% error rate. Instead of following a classical two-step approach in which a global null hypothesis is tested to begin with, and as a second step multiple

Table 9 Maintenance—significant results.

Pair	Lower	Upper	*p*-value
Improvement—Bug	0.009	0.029	9.067173e–04
New Feature—Bug	0.042	0.077	9.409836e–06
New Feature—Improvement	0.023	0.059	2.411054e–04
Task—Improvement	−0.085	−0.048	5.512420e–06
Sub-task—New Feature	−0.071	−0.028	2.764748e–04
Task—New Feature	−0.132	−0.083	1.791890e–07
Test—New Feature	−0.083	−0.007	2.128063e–02
Wish—New Feature	−0.116	−0.024	4.615626e–03
Task—Sub-task	−0.081	−0.036	1.468676e–04
Test—Task	0.025	0.101	2.563559e–03

comparisons are used to test sub-hypotheses related to each pair of groups, to answer our research question, we used the Tukey's test (*Tukey*, *1949*) which is a single-step multiple comparison procedure and statistical test.

To visualize the results obtained from the multiple contrast test procedure, we used the T∼ -graph presented by *Vasilescu et al.* (*2014a*). This visual comparison of multiple distributions using T∼-graphs provides an immediate understanding of groups located in higher positions in the graph, i.e., issue categories with higher politeness, while groups located in lower positions in the graph are related to issue categories with lower politeness. Other studies related to the application of T-graphs can be found in *Vasilescu, Filkov & Serebrenik* (*2013*) and *Vasilescu et al.* (*2014b*).

The approach used to build a T∼ -graph is the following (cf. *Vasilescu et al.* (*2014a*)):

- for each pair of groups it is necessary to analyse the 95% confidence interval to test if the corresponding null sub-hypothesis can be rejected;
- If the lower boundary of the interval is greater than zero for groups A and B, we conclude that the metric value is higher in A than in B;
- If the upper boundary of the interval is less than zero for groups A and B, we conclude that the metric value is lower in A than in B;
- If the lower boundary of the interval is less than zero and the upper boundary is greater than zero, we conclude that the data does not provide enough evidence to reject the null hypothesis.
- based on the results of the comparisons we construct the graph with nodes being groups and containing edges (A, B) if the metric value is higher in A than in B.

The result of the multiple contrast test procedure are presented in Tables 9 and 10. We used the *mctp* function for the Tukey's test, from the *nparcomp* R package (https://cran.r-project.org/web/packages/nparcomp/nparcomp.pdf) to obtain the tables. Table 9 summarises the significant results which we used to build a T∼ -graph. For the category *Brainstorming*, the upper boundary of the interval is less than zero and the upper boundary is greater than zero. In this situation the data does not provide enough evidence

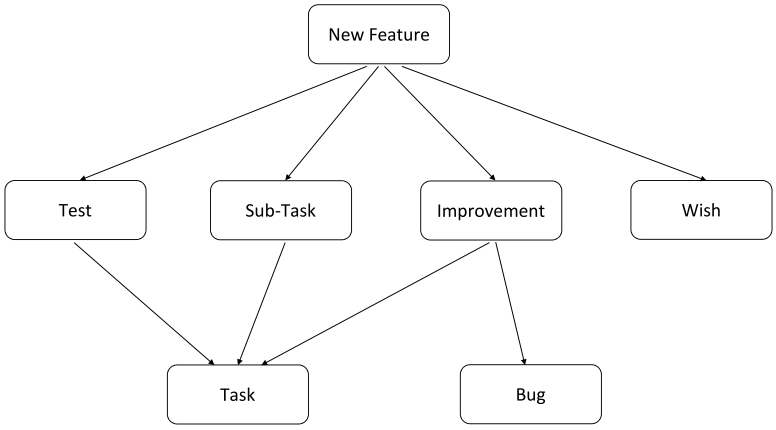


Figure 10 **T∼ -graph for issue maintenance.**

Table 10 **Issue priority classification.**

Pair	Lower	Upper	*p*-value
Critical—Blocker	−0.018	0.022	0.997
Major—Blocker	−0.070	−0.042	0
Minor—Blocker	−0.093	−0.065	0
Trivial—Blocker	−0.149	−0.113	0
Major—Critical	−0.073	−0.043	0
Minor—Critical	−0.097	−0.066	0
Trivial—Critical	−0.152	−0.115	0
Minor—Major	−0.030	−0.016	0
Trivial—Major	−0.088	−0.062	0
Trivial—Minor	−0.066	−0.039	0

and we cannot conclude that, for example, *Brainstorming* issue are more (or less) polite the *Bug* issues. Same thing happens for the category *Umbrella* and *New JIRA Project*.

Figure 10 shows the T∼ -graph resulting from the values in Table 9. The category *New Feature* is more polite than *Test, Sub-task, Improvement* and *Wish*; *Test, Sub-task* and *Improvement* are more polite than *Task*; *Improvement* are more polite than *Bug*. Issues with maintenance *Bug* are related to defects and software failures. This category presents the lower politeness. Issues with maintenance *New Feature* are proposals made by developers and it is interesting to see that when proposing something new, developers tend to be more polite. Issues on JIRA are also classified considering the level of priority, as *Major, Minor, Blocker* (e.g., an issue which blocks development and/or testing work), *Critical and Trivial*.

Table 10 shows the results of the multiple contrast test procedure for the different groups of issue priority. Figure 11 shows the associated T∼ -graph.

Blocker and *Critical* issues are more polite than *Major, Minor* and *Trivial* issues. *Major* issues are more polite than *Minor* and *Trivial*, while *Minor* are more polite than *Trivial*. *Trivial* issues are characterised by lower politeness.

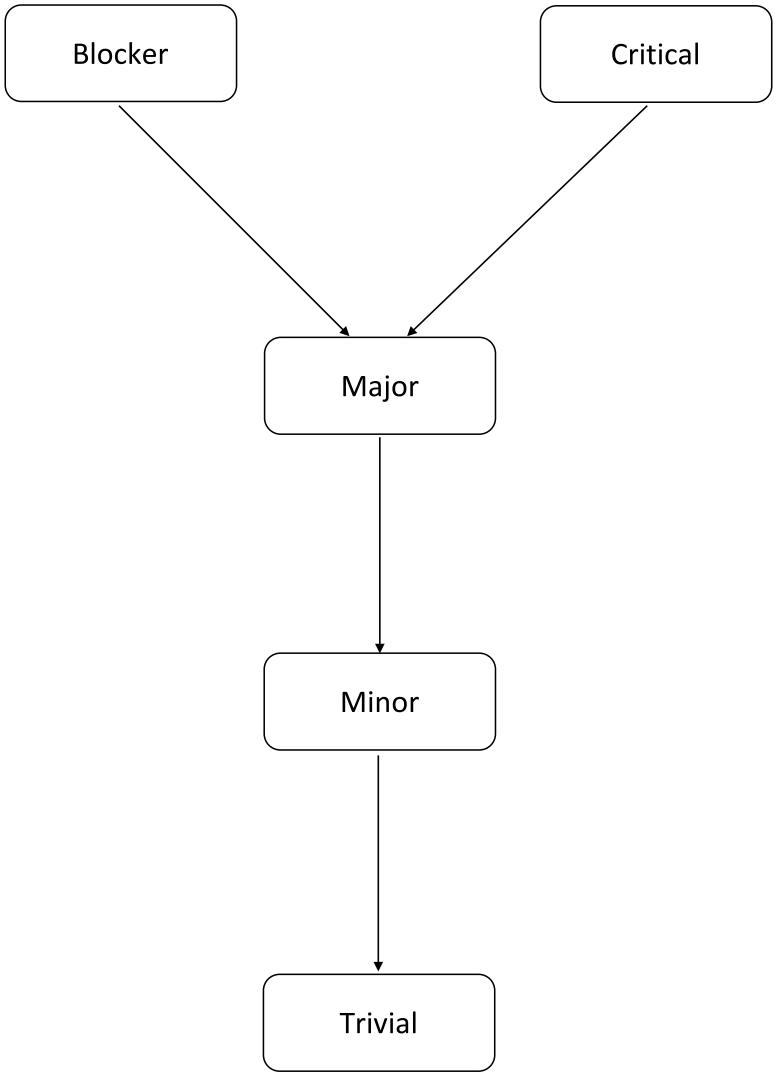


Figure 11 **T∼ -graph for issue Priority.**

Findings. *Comments related to issues with maintenance Bug, priority Minor and Trivial, tended to have a lower politeness. Issues with maintenance New Feature, priority Blocker and Critical, tended to have a higher politeness.*

DISCUSSION

Software development, as well as other fields, is an activity organised around team-based environments. The implementation of team structures is not simple and does not necessarily result in success, because it is not enough just to put people together in teams and to presume that everybody knows or agrees on what to do (*Allen & Hecht*, *2004*). People working together apply different personal assumptions and interpretations to their work tasks (*Keyton & Beck*, *2008*). Hence, conflicts within teams are possible. Conflicts affect teams' productivity and team leaders are certainly interested in knowing how to prevent,

avoid, or, in the worst case, manage conflicts which might occur. In this paper, we presented an analysis about the links between politeness and productivity of developers involved in a project. Politeness is a factor that certainly helps in diminishing conflict and friction between people. The findings of this study contribute in highlighting the importance and the impact of the psychological state of a developer on the software production process.

We started the paper by proposing that politeness information mined from software repositories like issue repositories could offer a way to investigate productivity during the software development process. We showed that issue fixing time for polite issues was shorter than issue fixing time for impolite and mixed-issues. This result matches common sense; if someone is asked to accomplish a task in a polite way, there is higher possibility for a relaxed collaboration and faster results. On the other hand, impolite requests can easily generate discomfort, stress and burnout, often negatively impacting the actual time taken to complete a given task. Surprisingly, mixed issues (commented with both polite and impolite comments) presented longer fixing time than impolite issues. The mixed interaction 'polite-impolite' between developers could explain this fact. *Ortu et al.* (*2016*) showed that when in the presence of impolite or negative comments, the probability of the next comment being impolite or negative was 13% and 25%, respectively. Hence part of the longer time could be spent in trying to shift the exchange of comments toward a (more) polite level. For example, in a small study of a single project with two deadlines (*Guzman*, *2013b*), the authors find that, as deadlines came closer, more and longer emails were exchanged with higher emotional intensity. The lower fixing time for impolite issues (compared to the fixing time of mixed issues) could also be related to the fact that developers (especially newcomers) being addressed with impolite comments can react faster because they feel emotionally pushed and want to show (to the community) that they are able to accomplish a task.

As a second point, we showed that a positive correlation existed between the percentage of polite comments and Magnetism and Stickiness of a project. For each project in the corpus, we first calculated the percentage of polite comments per month and the value of Magnetism and Stickiness, generating three time series. We studied each time series for stationarity and then performed correlation analysis. We found that the percentage of polite comments is, for the majority of the project in our corpus, positive correlated with Magnetism and Stickiness. However, we need to point out that the first cross correlation analysis performed for the non-stationary series presented weak correlation values (< 0.5). Higher correlation values were found after the cross correlation analysis of the stationary series (after differencing). The attractiveness of a project is indeed a complex phenomenon and there are different confounding factors (fame and importance of the project could be perceived also as a status-symbol, e.g., being part of the Linux developers community) of which politeness among developers can be part of. Further analysis on a larger corpus of projects are required. Our findings highlight the fact that politeness might be a factor affecting the attractiveness of a project.

Third, we found that the percentage of polite comments over time was (for the majority of the projects in our corpus) seasonal and not random. This is an interesting (and somewhat expected) fact that could help managers and developers in better understanding

the development process. Further experiments are required to better analyse the links between seasonality and deadlines that developers face during the development phases. Higher workload could lead to lower politeness, while normal activities can be linked with higher politeness.

Fourth, we studied how politeness varied with respect to JIRA maintenance types and issue priorities. Again, the results match common sense. *Bug* issues Maintenance were those with lower politeness. When something is broken and needs to be fixed, the situation is less attractive and could generate impolite reactions. On the contrary, *New Feature* issues Maintenance were the ones with higher politeness; those kind of issues indicate a new feature yet to be developed, hence, there might be higher enthusiasm among developers (a developer can be the first one who started working for a *New feature*; the level of freedom felt for the task can be higher leading to higher politeness) when dealing with such issues. Regarding issue *Priority*, *Critical* and *Blocker* issues were the categories with higher politeness, while *Trivial* issues were those characterised by lower politeness. Again, this is what one would expect, since critical issues are both important and challenging, while trivial issues might be related to minor programming mistakes and/or poor knowledge of programming practices

Knowing when lower politeness will occur can help managers in taking actions aimed at keeping the general mood high and relaxed, lowering and preventing conflicts, obtaining higher productivity as a result. The results presented in this study can be also helpful when defining a team of developers. Knowing the profile (from a politeness point of view) of the developers can provide hints for creating balanced teams. A JIRA plug-in able to present the politeness level of the communication flow in a graphical way (e.g., cockpit view) could help both developers and managers in constantly monitoring the general mood of the developers working for a company or for a project (in the case of open source collaboration paradigm).

THREATS TO VALIDITY

Threats to external validity correspond to the generalisation of our results (*Campbell & Stanley*, *1963*). In this study, we analysed comments from issue reports from 22 open source projects. Our results cannot be representative of all environments or programming languages, we considered only open-source systems and this could affect the generality of the study. Commercial software is usually developed using different platforms and technologies, by developers with different knowledge and background, with strict deadlines and cost limitations. Replication of this work on other open source systems and on commercial projects are needed to confirm our findings. Also, the politeness tool can be subject to bias due the domain used to train the machine learning classifier.

Threats to internal validity concern confounding factors that can influence the obtained results. Based on empirical evidence, we suppose a relationship between the emotional state of developers and what they write in issue reports (*Pang & Lee*, *2008*). Since the main goal of developer communication is the sharing of information, the consequence of removing or camouflaging emotions *may* make comments less meaningful and cause misunderstanding.

This work is focused on sentences written by developers for developers. To illustrate the influence of these comments, it is important to understand the language used by developers. We believe that the tool used for measuring politeness *Danescu-Niculescu-Mizil et al.* (*2013*) is valid in the software engineering domain, since the developers also used requests posted on Stack Overflow to train the classifier. The comments used in this study were collected over an extended period from developers unaware of being monitored. For this reason, we are confident that the emotions we analyzed were genuine. We do not claim any causality between politeness and the issue resolution time, but we built an explanatory model to understand the characteristics of issues with short and long fixing time. Confounds could have affected validity of the results for RQ1 about lower issue fixing time for polite and mixed issues. The number of developers involved in discussing issues might differ, as well as severity and complexity of an issue under analysis. Another threat to validity is related to classification of JIRA issue types. As highlighted by *Herzig, Just & Zeller* (*2013*) the categorisation of issue reports is dependent on the perspective of the observer. This fact could affect the results obtained for research question 4.

Threats to construct validity focus on how accurately the observations describe the phenomena of interest. The detection of emotions from issue reports presents difficulties due to vagueness and subjectivity. The politeness measures are approximated and cannot perfectly identify the precise context, given the challenges of natural language and subtle phenomena like sarcasm.

CONCLUSIONS AND FUTURE WORK

Software engineers have been trying to measure software to gain quantitative insights into its properties and quality since its inception. In this paper, we present the results about politeness and attractiveness on 22 open-source software projects developed using the Agile board of the JIRA repository. Our results show that the level of politeness in the communication process among developers does have an effect on both the time required to fix issues and the attractiveness of the project to both active and potential developers. The more polite developers were, the less time it took to fix an issue. In the majority of cases, the more the developers wanted to be part of project, the more they were willing to continue working on the project over time. This work is a starting point and further research on a larger number of projects is needed to validate our findings especially, considering proprietary software developed by companies, different programming languages and different dimension. The development of proprietary software follows different dynamics (e.g., strict deadlines and given budget) and this fact could lead to different results. We started the development of an application which will be able to automatically analyse all the comments on a issue tracking systems (as we have done for this paper) and will provide reports and data to managers, team-leaders and/or developers interested in understanding the health of a project from a ''mood'' point of view. The takeaway message is that politeness can only have a positive effect on a project and on the development process.

ACKNOWLEDGEMENTS

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions to improve the quality of the paper.

ADDITIONAL INFORMATION AND DECLARATIONS

Funding

The research presented in this paper was partly funded by the Engineering and Physical Sciences Research Council (EPSRC) of the UK under grant ref: EP/M024083/1. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Grant Disclosures

The following grant information was disclosed by the authors:
Engineering and Physical Sciences Research Council (EPSRC): EP/M024083/1.

Competing Interests

The authors declare there are no competing interests.

Author Contributions

- Giuseppe Destefanis, Marco Ortu and Steve Counsell conceived and designed the experiments, performed the experiments, analyzed the data, contributed reagents/materials/analysis tools, wrote the paper, prepared figures and/or tables, performed the computation work, reviewed drafts of the paper.
- Stephen Swift performed the experiments, analyzed the data, contributed reagents/materials/analysis tools, performed the computation work, reviewed drafts of the paper.
- Michele Marchesi contributed reagents/materials/analysis tools, reviewed drafts of the paper.
- Roberto Tonelli performed the experiments, contributed reagents/materials/analysis tools, performed the computation work, reviewed drafts of the paper.

Data Availability

The following information was supplied regarding data availability:

We used a public dataset available at http://openscience.us/repo/social-analysis/social-aspects.html, and all the scripts we prepared (along with the raw data) can be found at BitBucket: https://bitbucket.org/giuseppedestefanis/peerjcs_replicationpackage/

REFERENCES

Acuña ST, Gómez M, Juristo N. 2008. Towards understanding the relationship between team climate and software quality—a quasi-experimental study. *Empirical Software Engineering* **13(4)**:401–434 DOI 10.1007/s10664-008-9074-8.

Allen NJ, Hecht TD. 2004. The 'romance of teams': toward an understanding of its psychological underpinnings and implications. *Journal of Occupational and Organizational Psychology* **77(4)**:439–461 DOI 10.1348/0963179042596469.

Amabile T, Barsade SG, Mueller JS, Staw BM. 2005. Affect and creativity at work. *Administrative Science Quarterly* **50(3)**:367–403 DOI 10.2189/asqu.2005.50.3.367.

Banerjee A, Dolado JJ, Galbraith JW, Hendry D, et al. 1993. Co-integration, error correction, and the econometric analysis of non-stationary data. *OUP Catalogue*.

Bartels R. 1982. The rank version of von neumann's ratio test for randomness. *Journal of the American Statistical Association* **77(377)**:40–46 DOI 10.1080/01621459.1982.10477764.

Bazelli B, Hindle A, Stroulia E. 2013. On the personality traits of Stack Overflow users. In: *Software Maintenance (ICSM), 2013 29th IEEE International Conference on*. Piscataway: IEEE, 460–463.

Beck K, Beedle M, Van Bennekum A, Cockburn A, Cunningham W, Fowler M, Grenning J, Highsmith J, Hunt A, Jeffries R, Kern J. 2001. Manifesto for agile software development. *Available at http://www.agilemanifesto.org*.

Box GE, Pierce DA. 1970. Distribution of residual autocorrelations in autoregressive-integrated moving average time series models. *Journal of the American statistical Association* **65(332)**:1509–1526 DOI 10.1080/01621459.1970.10481180.

Brief AP, Weiss HM. 2002. Organizational behavior: affect in the workplace. *Annual Review of Psychology* **53(1)**:279–307 DOI 10.1146/annurev.psych.53.100901.135156.

Brockwell PJ, Davis RA. 2006. *Introduction to time series and forecasting*. New York: Springer Science & Business Media.

Campbell DT, Stanley JC. 1963. *Experimental and quasi-experimental designs for generalized causal inference*. Boston: Houghton Mifflin.

Capretz LF. 2003. Personality types in software engineering. *International Journal of Human-Computer Studies* **58(2)**:207–214 DOI 10.1016/S1071-5819(02)00137-4.

Cockburn A, Highsmith J. 2001. Agile software development: the people factor. *Computer* **11**:131–133.

Curtis B, Krasner H, Iscoe N. 1988. A field study of the software design process for large systems. *Communications of the ACM* **31(11)**:1268–1287 DOI 10.1145/50087.50089.

Danescu-Niculescu-Mizil C, Sudhof M, Jurafsky D, Leskovec J, Potts C. 2013. A computational approach to politeness with application to social factors. In: *Proceedings of ACL*.

Diebold FX, Rudebusch GD. 1991. On the power of Dickey-Fuller tests against fractional alternatives. *Economics Letters* **35(2)**:155–160 DOI 10.1016/0165-1765(91)90163-F.

Ehlers J. 2015. Socialness in the recruiting of software engineers. In: *Proceedings of the 12th ACM international conference on computing frontiers*. New York: ACM, 33.

Erez A, Isen AM. 2002. The influence of positive affect on the components of expectancy motivation. *Journal of Applied Psychology* **87(6)**:1055–1067 DOI 10.1037/0021-9010.87.6.1055.

Fagerholm F, Ikonen M, Kettunen P, Münch J, Roto V, Abrahamsson P. 2014. How do software developers experience team performance in lean and agile environments?

In: *Proceedings of the 18th international conference on evaluation and assessment in software engineering*. New York: ACM, 7.

Feldt R, Torkar R, Angelis L, Samuelsson M. 2008. Towards individualized software engineering: empirical studies should collect psychometrics. In: *Proceedings of the 2008 international workshop on Cooperative and human aspects of software engineering*. New York: ACM, 49–52.

Garcia D, Zanetti MS, Schweitzer F. 2013. The role of emotions in contributors activity: a case study on the Gentoo community. In: *Cloud and green computing (CGC), 2013 third international conference on*. Piscataway: IEEE, 410–417.

Gómez MN, Acuña ST, Genero M, Cruz-Lemus JA. 2012. How does the extraversion of software development teams influence team satisfaction and software quality? A controlled experiment. *International Journal of Human Capital and Information Technology Professionals (IJHCITP)* **3(4)**:11–24 DOI 10.4018/jhcitp.2012100102.

Graziotin D, Wang X, Abrahamsson P. 2014. Happy software developers solve problems better: psychological measurements in empirical software engineering. *PeerJ* **2**:e289 DOI 10.7717/peerj.289.

Graziotin D, Wang X, Abrahamsson P. 2015. How do you feel, developer? An explanatory theory of the impact of affects on programming performance. *PeerJ Computer Science* **1**:e18 DOI 10.7717/peerj-cs.18.

Guzman E. 2013a. Visualizing emotions in software development projects. In: *2013 1st IEEE working conference on software visualization—proceedings of VISSOFT 2013*. Piscataway: IEEE, 1–4.

Guzman E. 2013b. Visualizing emotions in software projects. In: *Software visualization (VISSOFT), 2013 first IEEE working conference on*. Piscataway: IEEE, 1–4.

Guzman E, Azócar D, Li Y. 2014. Sentiment analysis of commit comments in GitHub: an empirical study. In: *Proceedings of the 11th Working Conference on Mining Software Repositories—MSR 2014*. New York: ACM Press, 352–355.

Guzman E, Bruegge B. 2013. Towards emotional awareness in software development teams. In: *Proceedings of the 2013 9th joint meeting on foundations of software engineering*. New York: ACM Press, 671–674.

Hamilton JD. 1994. *Time series analysis*, vol. 2. Princeton university press Princeton.

Herzig K, Just S, Zeller A. 2013. It's not a bug, it's a feature: how misclassification impacts bug prediction. In: *Proceedings of the 2013 international conference on software engineering*. Piscataway: IEEE Press, 392–401.

Jongeling R, Datta S, Serebrenik A. 2015. Choosing your weapons: on sentiment analysis tools for software engineering research. In: *Software maintenance and evolution (ICSME), 2015 IEEE international conference on*. Piscataway: IEEE, 531–535.

Kaluzniacky E. 2004. *Managing psychological factors in information systems work: an orientation to emotional intelligence*. Hershey: IGI Global.

Keyton J, Beck SJ. 2008. Team attributes, processes, and values: a pedagogical framework. *Business Communication Quarterly* **71(4)**:488–504 DOI 10.1177/1080569908325863.

Konietschke F, Hothorn LA, Brunner E. 2012. Rank-based multiple test procedures and simultaneous confidence intervals. *Electronic Journal of Statistics* **6**:738–759 DOI 10.1214/12-EJS691.

Kramer ADI, Guillory JE, Hancock JT. 2014. Experimental evidence of massive-scale emotional contagion through social networks. *Proceedings of the National Academy of Sciences of the United States of America* **111(24)**:8788–8790 DOI 10.1073/pnas.1320040111.

Krištoufek L. 2014. Measuring correlations between non-stationary series with DCCA coefficient. *Physica A: Statistical Mechanics and its Applications* **402**:291–298 DOI 10.1016/j.physa.2014.01.058.

Kruskal WH, Wallis WA. 1952. Use of ranks in one-criterion variance analysis. *Journal of the American statistical Association* **47(260)**:583–621 DOI 10.1080/01621459.1952.10483441.

Kwiatkowski D, Phillips PC, Schmidt P, Shin Y. 1992. Testing the null hypothesis of stationarity against the alternative of a unit root: how sure are we that economic time series have a unit root? *Journal of econometrics* **54(1)**:159–178 DOI 10.1016/0304-4076(92)90104-Y.

Ljung GM, Box GE. 1978. On a measure of lack of fit in time series models. *Biometrika* **65(2)**:297–303 DOI 10.1093/biomet/65.2.297.

Miner AG, Glomb TM. 2010. State mood, task performance, and behavior at work: a within-persons approach. *Organizational Behavior and Human Decision Processes* **112(1)**:43–57 DOI 10.1016/j.obhdp.2009.11.009.

Murgia A, Concas G, Tonelli R, Ortu M, Demeyer S, Marchesi M. 2014a. On the influence of maintenance activity types on the issue resolution time. In: *Proceedings of the 10th international conference on predictive models in software engineering*. New York: ACM, 12–21.

Murgia A, Tourani P, Adams B, Ortu M. 2014b. Do developers feel emotions? an exploratory analysis of emotions in software artifacts. In: *Proceedings of the 11th working conference on mining software repositories*. New York: ACM, 262–271.

Novielli N, Calefato F, Lanubile F. 2014. Towards discovering the role of emotions in Stack Overflow. In: *Proceedings of the 6th international workshop on social software engineering*. New York: ACM, 33–36.

Ortu M, Adams B, Destefanis G, Tourani P, Marchesi M, Tonelli R. 2015a. Are bullies more productive? Empirical study of affectiveness vs. issue fixing time. In: *Proceedings of the 12th working conference on mining software repositories, MSR 2015*.

Ortu M, Destefanis G, Counsell S, Swift S, Tonelli R, Marchesi M. 2016. Arsonists or firefighters? Affectiveness in agile software development. In: *Agile processes, in software engineering, and extreme programming*. Berlin Heidelberg: Springer International Publishing, 144–155.

Ortu M, Destefanis G, Kassab M, Counsell S, Marchesi M, Tonelli R. 2015b. Would you mind fixing this issue? an empirical analysis of politeness and attractiveness in software developed using agile boards. In: *Agile processes, in software engineering,*

and extreme programming. Berlin Heidelberg: Springer International Publishing, 129–140.

Ortu M, Destefanis G, Kassab M, Marchesi M. 2015c. Measuring and understanding the effectiveness of jira developers communities. In: *Proceedings of the 6th International Workshop on Emerging Trends in Software Metrics, WETSoM 2015*.

Ortu M, Destefanis G, Murgia A, Marchesi M, Tonelli R, Adams B. 2015d. The JIRA repository dataset: understanding social aspects of software development. In: *Proceedings of the 11th international conference on predictive models and data analytics in software engineering*. New York: ACM, 1.

Pang B, Lee L. 2008. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval* **2(1–2)**:1–135 DOI 10.1561/1500000011.

Panichella S, Di Sorbo A, Guzman E, Visaggio CA, Canfora G, Gall HC. 2015. How can I improve my app? Classifying user reviews for software maintenance and evolution. In: *Software maintenance and evolution (ICSME), 2015 IEEE international conference on*. Piscataway: IEEE, 281–290.

Pennebaker JW, Francis ME, Booth RJ. 2001. Linguistic inquiry and word count: Liwc 2001. *Mahway: Lawrence Erlbaum Associates* **71**:2001.

Perry T. 2008. Drifting toward invisibility: the transition to the electronic task board. In: *Agile, 2008. AGILE'08. Conference*. Piscataway: IEEE, 496–500.

Pletea D, Vasilescu B, Serebrenik A. 2014. Security and emotion: sentiment analysis of security discussions on GitHub. In: *Proceedings of the 11th working conference on mining software repositories*. New York: ACM, 348–351.

Priestley MB. 1981. *Spectral analysis and time series*. San Diego: Academic Press.

Priestley MB. 1988. *Non-linear and non-stationary time series analysis*. Amsterda: Elsevier.

R Development Core Team. 2014. *R: a language and environment for statistical computing*. Vienna: the R Foundation for Statistical Computing. *Available at http://www.R-project.org/*.

Rigby PC, Hassan AE. 2007. What can OSS mailing lists tell us? A preliminary psychometric text analysis of the Apache developer mailing list. In: *Proceedings of the fourth international workshop on mining software repositories*. Piscataway: IEEE Computer Society, 23.

Roberts JA, Hann I-H, Slaughter SA. 2006. Understanding the motivations, participation, and performance of open source software developers: a longitudinal study of the Apache projects. *Management Science* **52(7)**:984–999 DOI 10.1287/mnsc.1060.0554.

Rousinopoulos A-I, Robles G, González-Barahona JM. 2014. Sentiment analysis of free/open source developers: preliminary findings from a case study. *Revista Eletrônica de Sistemas de Informação* **13(2)**:1677–3071 DOI 10.5329/RESI.

Said SE, Dickey DA. 1984. Testing for unit roots in autoregressive-moving average models of unknown order. *Biometrika* **71(3)**:599–607 DOI 10.1093/biomet/71.3.599.

Siegel S. 1956. *Nonparametric statistics for the behavioral sciences*. New York: McGraw-Hill.

Śliwerski J, Zimmermann T, Zeller A. 2005. Don't program on Fridays! How to locate fix-inducing changes. In: *Proceedings of the 7th workshop software reengineering*. *Available at http://thomas-zimmermann.com/publications/files/sliwerski-wsr-2005.pdf*.

Steinmacher I, Conte TU, Gerosa M, Redmiles D. 2015. Social barriers faced by newcomers placing their first contribution in open source software projects. In: *Proceedings of the 18th ACM conference on computer supported cooperative work & social computing*. New York: ACM, 1–13.

Tan S, Howard-Jones P. 2014. Rude or polite: do personality and emotion in an artificial pedagogical agent affect task performance? In: *2014 global conference on teaching and learning with technology (CTLT 2014)*. 41.

Tourani P, Jiang Y, Adams B. 2014. Monitoring sentiment in open source mailing lists: exploratory study on the Apache ecosystem. In: *Proceedings of 24th annual international conference on computer science and software engineering*. Armonk: IBM Corporation, 34–44. *Available at http://mcis.soccerlab.polymtl.ca/publications/2014/cascon14.pdf*.

Tsay J, Dabbish L, Herbsleb J. 2014. Let's talk about it: evaluating contributions through discussion in GitHub. In: *Proceedings of the 22nd ACM SIGSOFT international symposium on foundations of software engineering*. New York: ACM, 144–154.

Tukey JW. 1949. Comparing individual means in the analysis of variance. *Biometrics* **5(2)**:99–114 DOI 10.2307/3001913.

Vasilescu B, Filkov V, Serebrenik A. 2013. Stack overflow and GitHub: associations between software development and crowdsourced knowledge. In: *Social computing (SocialCom), 2013 international conference on*. Piscataway: IEEE, 188–195.

Vasilescu B, Serebrenik A, Goeminne M, Mens T. 2014a. On the variation and specialisation of workload—a case study of the Gnome ecosystem community. *Empirical Software Engineering* **19(4)**:955–1008 DOI 10.1007/s10664-013-9244-1.

Vasilescu B, Serebrenik A, Mens T, Van den Brand MGJ, Pek E. 2014b. How healthy are software engineering conferences? *Science of Computer Programming* **89**:251–272 DOI 10.1016/j.scico.2014.01.016.

VersionOne. 2013. 8th Annual State of Agile Survey report. San Francisco: VersionOne. *Available at https://www.versionone.com/pdf/2013-state-of-agile-survey.pdf*.

Weiss C, Premraj R, Zimmermann T, Zeller A. 2007. How long will it take to fix this bug? In: *Proceedings of the fourth international workshop on mining software repositories*. Piscataway: IEEE Computer Society, 1.

Winschiers H, Paterson B. 2004. Sustainable software development. In: *Proceedings of the 2004 annual research conference of the South African Institute of Computer Scientists and Information Technologists on IT research in developing countries*. South African Institute for Computer Scientists and Information Technologists, 274–278.

Yamashita K, Kamei Y, McIntosh S, Hassan AE, Ubayashi N. 2016. Magnet or sticky? Measuring project characteristics from the perspective of developer attraction and retention. *Journal of Information Processing* **24(2)**:339–348 DOI 10.2197/ipsjjip.24.339.

Yamashita K, McIntosh S, Kamei Y, Ubayashi N. 2014. Magnet or sticky? An OSS project-by-project typology. In: *Proceedings of the 11th working conference on mining software repositories*. New York: ACM, 344–347.

Zhang H, Gong L, Versteeg S. 2013. Predicting bug-fixing time: an empirical study of commercial software projects. In: *Proceedings of the 2013 international conference on software engineering*. Piscataway: IEEE Press, 1042–1051.

Zhang F, Khomh F, Zou Y, Hassan AE. 2012. An empirical study on factors impacting bug fixing time. In: *2012 19th working conference on reverse engineering (WCRE)*. Piscataway: IEEE, 225–234.